Package: algaeClassify

Title: Determine Phytoplankton Functional Groups Based on Functional Traits

- release v2.0.0
  - a newer version of this software package may be available.
    - provisional updates: https://code.usgs.gov/asc/algaeClassify/-/tree/main
    - approved releases: https://code.usgs.gov/asc/algaeClassify/-/releases

Authors

Vijay Patil (ORCID: 0000-0002-9357-194X) U.S. Geological Survey - Alaska Science Center
Torsten Seltmann
Nico Salmaso
Orlane Anneville
Marc Lajeunesse
Dietmar Straile

Suggested Citation

Patil, V.P., Seltmann, T., Salmaso, N., Anneville, O., Lajeunesse, M., Straile, D., 2023. algaeClassify (ver 1.0.0, Month 2023): U.S. Geological Survey software release, https://doi.org/10.5066/F7S46Q3F

Contact

Vijay Patil vpatil@usgs.gov - U.S. Geological Survey - Alaska Science Center; 4210 University Drive; Anchorage, Alaska 99508 USA; 907-786-7000 ascweb@usgs.gov

Software Requirements

- Requires R (version 4.2.0 or later)
  - available for free download from the Comprehensive R Archive Network (CRAN) https://cran.r-project.org

Distribution

- Comprehensive R Archive Network (CRAN) https://cran.r-project.org
- The U.S. Geological Survey through this repository https://doi.org/10.5066/F7S46Q3F

Package Overview

The goal of algaeClassify is to facilitate the analysis of taxonomic and functional trait data for phytoplankton.

Functions that facilitate the use of accepted taxonomic nomenclature, collection of functional trait data, and assignment of functional group classifications to phytoplankton species. Possible classifications include Morpho-functional group (MFG; [Salmaso et al. 2015] doi:10.1111/fwb.12520) and CSR, based on [Reynolds 1988] (Functional morphology and the adaptive strategies of phytoplankton. In C.D. Sandgren (ed). Growth and reproductive strategies of freshwater phytoplankton, 388-433. Cambridge University Press, New York) and [Reynolds 2006] https://doi.org/10.1017/CBO9780511542145.

Versions 2.0.0 and later includes new functions for querying the algaebase online taxonomic database (www.algaebase.org), however these functions require a valid API key that must be acquired from the algaebase admin. Note that none of the algaeClassify authors are affiliated with algaebase in any way. Taxonomic names can also be checked against a variety of taxonomic databases using the geographic name resolution service (GNRS) via wrapper functions for the R taxize package, with convenient output format and unlikely names for phytoplankton taxa removed. In addition, currently accepted and outdated synonyms, and higher taxonomy, can be extracted for lists of species from the ITIS database using wrapper functions for the R ritis package. The algaeClassify package is a product of the GEISHA (Global Evaluation of the Impacts of Storms on freshwater Habitat and Structure of phytoplankton Assemblages), funded by CESAB (Centre for Synthesis and Analysis of Biodiversity) and the USGS John Wesley Powell Center for Synthesis and Analysis, with data and other support provided by members of GLEON (Global Lake Ecology Observation Network).

Installation

To use the new algaebase search functions, you MUST install v2.0.0 from github with:

```
require(devtools)
# install_git("https://code.usgs.gov/asc/algaeClassify",ref="master")
#this works for gitlab code.usgs.gov
install_github("vppatil/algaeClassifyV2",ref="main",dependencies=TRUE)
#install v2.0.0 from github
```

Next, load the package and ensure you have the correct version installed.

```
library(algaeClassify)
citation("algaeClassify")
```

##April 5, 2023: New functions for querying algaeBase (www.algaebase.org)!!!! Algaebase search function examples

```
#check out the package
help(package="algaeClassify")

#View the new algaebase search functions:
help("algaebase_species_search")
help("algaebase_genus_search")
help("algaebase_search_df")
```

##Using your API key

The Algaebase functions require an API key. You can obtain one from [Algaebase] https://www.algaebase.org/api.

There are several options for using your api key. 1) assigning it to an R object, and using it in function calls

```
apikey<- "asasdfasdfasdfasfd" (not a real key)
algaebase_genus_search(genus="Anabaena",apikey=apikey)
```

2) Saving it in a text file (e.g. "keyfile.txt"). You will need to give the filename and path using the api_file function argument.

```
algaebase_genus_search(genus="Anabaena",api_file="keyfile.txt")
```

3) Finally, you can set your key as an environment variable. To do so, open or create a .Renviron text file in your home directory. One way to do this is by running the following line:

```
file.edit("~/.Renviron")
```

Add a line to the .Renviron file like the following, but use your actual key after the = symbol:

ALGAEBASE_APIKEY=yourKeyHere

Finally, save and close the file, then restart R for changes to take effect. Once the ALGAEBASE_APIKEY variable is defined, you do not need to specify it in the algaebase search functions.

Algaebase search function examples:

#You can search for a single genus

```
?algaebase_genus_search
algaebase_genus_search("Anabaena")
```

#Or a genus and species name

```
algaebase_species_search("Anabaena","flos-aquae")

#There are several arguments for these functions.
#you can control whether to include higher taxonomy in the output
algaebase_genus_search(genus="Navicula",higher=TRUE)

#You can also choose to include the full species name with author and date
algaebase_genus_search(genus="Navicula",higher=TRUE,long=TRUE)

#The default only returns exact matches and the most recent entry in algaebase,
#but you can override that behavior:

algaebase_species_search(genus="Nitzschia",species="acicularis",
newest.only=FALSE,exact.matches.only=FALSE,long=TRUE)
```

In all cases, the output will return the currently accepted name, as well as the name that was supplied by the user. There are columns indicating whether the input name is currently accepted and whether an exact match was found.

If desired, you can view the raw output in JSON format

```
algaebase_genus_search(genus="Cyclotella",higher=TRUE,print.full.json=TRUE)
```

##Search a list of names: Finally, you can submit a data.frame of phytoplankton names to algaebase the data frame should have columns named genus and species

This will only return 1 result per name. If there are no exact matches it will return NA If there is no match for genus+species, it will search for a genus-only match or you can specify genus.only searches for the entire dataset.

```
data(lakegeneva) #load small example dataset
head(lakegeneva) #view example dataset

lakegeneva<-genus_species_extract(lakegeneva,phyto.name="phyto_name")
lakegeneva<-lakegeneva[!duplicated(lakegeneva$phyto_name),]
lakegeneva.algaebase<-algaebase_search_df(lakegeneva,higher=TRUE,
genus.name="genus",species.name="species")
head(lakegeneva.algaebase)
```

##Other taxonomic search functions. Version 2.0.0 includes functions for searching the ITIS database and for using the Global Names Resolver (GNR). These functions are based on the **ritis** and **taxize** packages, respectively.

```
help("genus_search_itis")
help("species_search_itis")
help("itis_search_df")

#ITIS
genus_search_itis(genus="Mougeotia",higher=TRUE)
```

```
species_search_itis(genspp="Anabaena flos-aquae")

#GNR (Global Names Resolver)
#GNR/taxize use fuzzy/partial matching, and search multiple databases.
#If you do not find a hit in algaebase or ITIS, you can try searching for a
#partial match via GNR

help("gnr_simple")
help("gnr_simple_df")

name<-"Anabaena flos-aquae"
gnr_simple(name=name,sourceid=3) #Search ITIS
gnr_simple(name=name,sourceid=NULL) #search for matches from any source
```

##Examples of other algaeClassify functions: This is a basic example which shows you how to use algaeClassify to 1) identify anomalies in a time-series of phytoplankton species 2) calculate aggregate abundance at a higher taxonomic level (genus) 3) re-plot species accumulation curves to see if the taxonomic standardization and aggregation to higher taxonomy have resolved the anomalies.

```
library(algaeClassify)

data(lakegeneva) #load a demonstration dataset

#view species accumulation curve over duration of dataset to check for anomalies
accum(lakegeneva,phyto_name='genus',column='biovol_um3_ml',n=100,
datename='date_dd_mm_yy',dateformat='%d-%m-%y')

#clean up binomial names and extract genus and species to new columns
lakegeneva<-genus_species_extract(lakegeneva,phyto.name='phyto_name')

#aggregate abundance data to genus level
lakegeneva.genus<-phyto_ts_aggregate(lakegeneva,SummaryType='abundance',
AbundanceVar='biovol_um3_ml',GroupingVar1='genus')

#plot accumulation curve again, but at genus level
accum(lakegeneva.genus,phyto_name='genus',column='biovol_um3_ml',n=100,
datename='date_dd_mm_yy',dateformat='%Y-%m-%d')

#assign species to morphofunctional groups using species names
lakegeneva.mfg<-species_to_mfg_df(lakegeneva)

#plot accumulation curve again, but at genus level
accum(lakegeneva.genus,phyto_name='genus',column='biovol_um3_ml',n=100,
      datename='date_dd_mm_yy',dateformat='%Y-%m-%d')

#classify taxa to CSR and visualize relative abundance of CSR groups by month
data(lakegeneva)

lakegeneva<-genus_species_extract(lakegeneva,phyto.name='phyto_name')
lakegeneva<-species_to_mfg_df(lakegeneva)
lakegeneva<-mfg_csr_convert_df(lakegeneva,mfg='MFG')
csrAbundance.by.month<-date_mat(lakegeneva,abundance.var='biovol_um3_ml',
                                taxa.name='CSR',time.agg='month')
```

```
#make a simple heatmap of mean daily csr group abundance by month
heatmap(csrAbundance.by.month,Rowv=NA,Colv=NA,col=viridis(10))
```